

Towards More Practical Linear Programming-based Techniques for Algorithmic Mechanism Design^{*}

Khaled Elbassioni¹, Kurt Mehlhorn², and Fahimeh Ramezani³

¹ Masdar Institute of Science and Technology, Abu Dhabi, UAE

² Max Planck Institute for Informatics, Campus E1 4, 66123, Saarbrücken, Germany

³ Department of Mathematics, University of Isfahan, Isfahan 81746-73441, Iran
kelbassioni@masdar.ac.ae, mehlhorn@mpi-inf.mpg.de, f.ramezani@sci.ui.ac.ir

Abstract. R. Lavi and C. Swamy (FOCS 2005, J. ACM 2011) introduced a general method for obtaining truthful-in-expectation mechanisms from linear programming based approximation algorithms. Due to the use of the Ellipsoid method, a direct implementation of the method is unlikely to be efficient in practice. We propose to use the much simpler and usually faster multiplicative weights update method instead. The simplification comes at the cost of slightly weaker approximation and truthfulness guarantees.

1 Introduction

Algorithmic mechanism design is the art of designing and implementing the rules of a game to achieve a desired outcome from a set of possible outcomes. Each player (agent) has a valuation that assigns a value to each possible outcome. The desired outcome is the one that maximizes the sum of the valuations; this sum is usually called *social welfare*. The players are assumed to be selfish: they report valuations to the mechanism, which may differ from the true valuations. Players may lie about their valuations in order to direct the mechanism into an outcome favorable to them. The mechanism computes an outcome and payments for the players. The *utility of a player* is her/his value of the outcome computed by the mechanism minus her/his payment charged by the mechanism. The agents are interested in optimizing their personal utility. Social welfare and personal utilities are determined with respect to the true valuations of the players, although they are not public knowledge. The purpose of the payments is to incentivize the players to report their true valuations. A mechanism is *truthful* if reporting the truth is a best strategy for each player irrespective of the inputs provided by the other players. A mechanism is *efficient* if the outcome and the payments can be computed in polynomial time. The *underlying optimization problem* is the computation of an outcome maximizing social welfare given the valuations of the players.

If the underlying optimization problem can be efficiently solved to optimality, the celebrated VCG mechanism (see, e.g., [NRTV07]) achieves truthfulness, social welfare optimization, and polynomial running time. The computation of the outcome and the computation of the payments requires to solve the underlying optimization problem to optimality.

Many optimization problems are NP-hard and hence are unlikely to have an exact algorithm with polynomial running time. However, it might be possible to solve the problem approximately in polynomial running time.

An example is the combinatorial auction problem. There is a set of m items to be sold to a set of n players. The (reported) value of a set S of items to the i -th player is $v_i(S)$ with $v_i(\emptyset) = 0$ and $v_i(S) \leq v_i(T)$ whenever $S \subseteq T$. Let $x_{i,S}$ be a 0-1 variable indicating that set S is given to player i . Then $\sum_S x_{i,S} \leq 1$ for every player i as at most one set can be given to i , and $\sum_i \sum_{S:j \in S} x_{i,S} \leq 1$ for every item j as any item can be given away only once. The social welfare is $\sum_{i,S} v_i(S) x_{i,S}$. The linear programming relaxation is obtained by replacing the integrality constraints for $x_{i,S}$ by $0 \leq x_{i,S} \leq 1$. Note that the number d of variables is exponential in the number of items, namely $d = n2^m$. The linear program is of the packing type, i.e., if x is feasible and $y \leq x$, then y is feasible. For the combinatorial auction problem, $O(\sqrt{n})$ -approximation algorithms exist and these

^{*} A preliminary version of these results was presented at SAGT 2015 [EMR15].

algorithms also provide the corresponding integrality-gap-verifier (the definition is given below) with $\alpha = 1/\sqrt{n}$ ([BKV05,KS98,Rag88]).

For many integer linear programming problems, approximation algorithms are known that first solve the corresponding linear programming relaxation and then construct an integral solution either by rounding or by primal-dual methods. Lavi and Swamy ([LS05,LS11]) showed that certain linear programming based approximation algorithms for the social welfare problem can be turned into randomized mechanisms that are truthful-in-expectation, i.e., reporting the truth maximizes the expected utility of a player. The LS-mechanism is powerful (see [LS05,LS11,CEF10,HKV11] for applications), but unlikely to be efficient in practice because of its use of the Ellipsoid method. *We show how to use the multiplicative weights update method instead. This results in simpler algorithms at the cost of somewhat weaker approximation and truthfulness guarantees.*

We next review the LS-mechanism. It applies to integer linear programming problems of the packing type for which the linear programming relaxation can be solved exactly and for which an α -integrality gap verifier is available. More precisely:

1. Let $\mathcal{Q} \subseteq \mathbb{R}_{\geq 0}^d$ be a *packing polytope*, i.e., \mathcal{Q} is a bounded convex polytope contained in the non-negative orthant of d -dimensional space with the property that if $y \in \mathcal{Q}$ and $x \leq y$ then $x \in \mathcal{Q}$. The linear programming problem for \mathcal{Q} asks to find for a given d -dimensional vector v a point $x^* = \operatorname{argmax}_{x \in \mathcal{Q}} v^T x$.
2. We use $\mathcal{Q}_{\mathcal{I}} := \mathcal{Q} \cap \mathbb{Z}^d$ for the set of integral points in \mathcal{Q} . The integer linear programming problem for $\mathcal{Q}_{\mathcal{I}}$ asks to find for a given d -dimensional vector v a point $x^* = \operatorname{argmax}_{x \in \mathcal{Q}_{\mathcal{I}}} v^T x$. We use $x^1, x^2, \dots, x^j, \dots$ to denote the elements of $\mathcal{Q}_{\mathcal{I}}$ and \mathcal{N} for the index set of all elements in $\mathcal{Q}_{\mathcal{I}}$.
3. An α -integrality-gap-verifier for $\mathcal{Q}_{\mathcal{I}}$ for some $\alpha \in (0, 1]$ is an efficient algorithm that on input $v \in \mathbb{R}^d$ and $x^* \in \mathcal{Q}$, returns an $x \in \mathcal{Q}_{\mathcal{I}}$ such that

$$v^T x \geq \alpha v^T x^*.$$

The mechanism consists of three main steps:

1. Let $v_i \in \mathbb{R}_{\geq 0}^d, 1 \leq i \leq n$, be the reported valuation of the i -th player and let $v = \sum_i v_i$ be the accumulated reported valuation. Solve the LP-relaxation, i.e., find a maximizer $x^* = \operatorname{argmax}_{x \in \mathcal{Q}} v^T x$ for the social welfare of the fractional problem, and determine the VCG prices⁴ p_1, \dots, p_n . The allocation x^* and the VCG-prices are a truthful mechanism for the fractional problem.
2. Write $\alpha \cdot x^*$ as a *convex combination of integral solutions* in \mathcal{Q} , i.e., $\alpha \cdot x^* = \sum_{j \in \mathcal{N}} \lambda_j x^j$, $\lambda_j \geq 0$, $\sum_{j \in \mathcal{N}} \lambda_j = 1$, and $x^j \in \mathcal{Q}_{\mathcal{I}}$. This step requires the α -integrality-gap-verifier.
3. Pick the integral solution x^j with probability λ_j , and charge the i -th player the price $p_i \cdot (v_i^T x^j / v_i^T x^*)$. If $v_i^T x^* = 0$, charge zero.

The LS-mechanism approximates social welfare with factor α (is α -socially efficient) and guarantees truthfulness-in-expectation, i.e., it converts a truthful fractional mechanism into an α -approximate truthful-in-expectation integral mechanism. With respect to practical applicability, steps 1 and 2 are the two major bottlenecks. Step 1 requires solving $n + 1$ linear programs, one for the fractional solution and one for each price; an exact solution requires the use of the Ellipsoid method (see e.g. [GLS88]), if the dimension is exponential. Furthermore, up to recently, the only method known to perform the decomposition in Step 2 is through the Ellipsoid method. An alternative method avoiding the use of the Ellipsoid method was recently given by Kraft, Fadaei, and Bichler [KFB14]. We comment on their result in the next section.

1.1 Our Results

Our result concerns the design and analysis of a practical algorithm for the LS-scheme. We first consider the case where the LP-relaxation of SWM (social welfare maximization) in Step 1 of the

⁴ $p_i = \sum_{j \neq i} v_j^T (\hat{x} - x^*)$, where $\hat{x} = \operatorname{argmax}_{x \in \mathcal{Q}} \sum_{j \neq i} v_j^T x$.

LS-scheme can be solved exactly and efficiently and then our problem reduces to the design of a practical algorithm for Step 2. Afterwards, we consider the more general problem where only an FPTAS for the LP-relaxation is available.

Convex Decomposition. Over the past 15 years, simple and fast methods have been developed for solving packing and covering linear programs [BI06,GK95,GK07,Kha04,KY07,PST91,You01] within an arbitrarily small error guarantee ε . These methods are based on the multiplicative weights update (MWU) method [AHK12], in which a very simple update rule is repeatedly performed until a near-optimal solution is obtained. We show how to replace the use of the Ellipsoid method in Step 2 by an approximation algorithm for covering linear programs. This result is the topic of Section 2.

Theorem 1. *Let $\varepsilon > 0$ be arbitrary. Given a fractional point $x^* \in \mathcal{Q}$, and an α -integrality-gap verifier for $\mathcal{Q}_{\mathcal{I}}$, we can find a convex decomposition*

$$\frac{\alpha}{1+4\varepsilon} \cdot x^* = \sum_{j \in \mathcal{N}} \lambda_j x^j.$$

The convex decomposition has size (= number of nonzero λ_j) at most $s(1 + \lceil \varepsilon^{-2} \ln s \rceil)$, where s is the size of the support of x^ (= number of nonzero components). The algorithm makes at most $s \lceil \varepsilon^{-2} \ln s \rceil$ calls to the integrality-gap-verifier.*

Kraft, Fadaei, and Bichler [KFB14] obtained a related result independently. However, their construction is less efficient in two aspects. First, it requires $O(s^2 \varepsilon^{-2})$ calls of the integrality-gap-verifier. Second, the size of their convex decomposition might be as large as $O(s^3 \varepsilon^{-2})$. In the combinatorial auction problem, $s = n + m$. Theorem 1 together with Steps 1 and 3 of the LS scheme implies a mechanism that is truthful-in-expectation and has $(\alpha/(1+4\varepsilon))$ -social efficiency.

We leave it as an open problem whether the quadratic dependency of the size of the decomposition on ε can be improved⁵.

Approximately Truthful-in-Expectation Mechanism. We drop the assumption that the fractional SWM-problem can be solved optimally and assume instead that we have an FPTAS for it. We assume further that the problem is *separable*, which means that the variables can be partitioned into disjoint groups, one for each player, such that the value of an allocation for a player depends only on the variables in his group, i.e.,

$$v_i(x) = v_i(x_i),$$

where x_i is the set of variables associated⁶ with player i . Formally, any outcome $x \in \mathcal{Q} \subseteq \mathbb{R}^d$ can be written as $x = (x_1, \dots, x_n)$ where $x_i \in \mathbb{R}^{d_i}$ and $d = d_1 + \dots + d_n$. We further assume that for each player $i \in [n]$, there is a *dominating allocation* $u^i \in \mathcal{Q}$ that maximizes his value for every valuation v_i , i.e.,

$$v_i(u^i) = \max_{z \in \mathcal{Q}} v_i(z), \tag{1}$$

for every $v_i \in \mathcal{V}_i$, where \mathcal{V}_i denotes the possible valuations of player i . For the case of a combinatorial auction, the allocation u^i allocates all items to player i .

Theorem 2. *Let $\varepsilon_0 \in (0, 1/2]$. Define $\varepsilon = \Theta(\frac{\varepsilon_0^5}{n^4})$. Assuming that the fractional SWM-problem has an FPTAS, is separable, and has a dominant allocation for every player i , and that there is an α -integrality gap verifier for $\mathcal{Q}_{\mathcal{I}}$, there is a polynomial time randomized integral mechanism with the following properties:*

⁵ We remark that recent progress [ZO15,WRM15] on solving LPs of the packing/covering type has resulted in an almost *linear* dependence of the running time on $\frac{1}{\varepsilon}$. However, the current methods do not work in the oracle model and hence cannot be directly applied in our setting.

⁶ In the combinatorial auction problem, x_i comprises all variables $x_{i,S}$. The value of an allocation x for player i is given by $\sum_S v_i(S) x_{i,S}$.

- (C1) No positive transfer, i.e., prices are nonnegative.
- (C2) Individually rational with probability $1 - \varepsilon_0$, i.e., the utility of any truth-telling player is non-negative with probability at least $1 - \varepsilon_0$.
- (C3) $(1 - \varepsilon_0)$ -truthful-in-expectation, i.e., reporting the truth maximizes the expected utility of a player up to a factor $1 - \varepsilon_0$.
- (C4) γ -socially efficient, where $\gamma = \alpha(1 - \varepsilon)(1 - \varepsilon_0)/(1 + 4\varepsilon)$.

Our mechanism is based on constructing a randomized fractional mechanism with properties (C1) to (C3) and being $(1 - \varepsilon)(1 - \varepsilon_0)$ -socially efficient and then converting the mechanism into an integral mechanism with the properties above. The conversion is simple. Let us assume that x is a fractional allocation obtained from the fractional mechanism. We apply our convex decomposition technique and Step 3 of the Lavi-Swamy mechanism to obtain an integral randomized mechanism that satisfies (C1) to (C4). We show this result in Section 3.

Our fractional mechanism refines the one given in [DRVY11], where the dependency of ε on n and ε_0 is as $\varepsilon = \Theta(\varepsilon_0/n^9)$. A recent experimental study of our mechanism on Display Ad Auctions [EJ15] shows the applicability of our techniques in practice.

We leave it as an open problem whether the dependency of ε on ε_0 and n can be improved.

On the Existence of an FPTAS for the Fractional SWM-Problem. We close the survey of our results with a comment on the existence of an FPTAS for the fractional SWM-problem. Consider a packing linear program

$$\max c^T x \quad \text{subject to} \quad Ax \leq b, \quad x \geq 0,$$

where $A \in \mathbb{R}_{\geq 0}^{m \times n}$ is an $m \times n$ matrix with non-negative entries and $c \in \mathbb{R}_{> 0}^n$, $b \in \mathbb{R}_{> 0}^m$ are positive vectors. We may assume that each column of A contains a non-zero entry as otherwise the problem is trivially unbounded. For every $\kappa \geq 1$ and weight vector $z \in \mathbb{R}_{\geq 0}^m$, let $\mathcal{O}_\kappa(z)$ denote a κ -approximation oracle that returns a j such that

$$\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i} \leq \kappa \cdot \min_{j' \in [n]} \frac{1}{c_{j'}} \sum_{i=1}^m \frac{z_i a_{ij'}}{b_i}.$$

Garg and Könemann [GK07] presented an algorithm that uses the oracle \mathcal{O}_κ to construct an approximation with a factor arbitrarily close to $1/\kappa$. For $\kappa = 1$, their algorithm is an FPTAS.

What is the approximation oracle in case of the combinatorial auction problem? In this problem, we have one constraint for each player and one constraint for each item. Let $y_i \geq 0$ be the weight for agent i and $z_j \geq 0$ be the weight for item j . Then oracle $\mathcal{O}_1(y, z)$ requires to find the pair

$$(i, S) := \operatorname{argmin}_{(k, T)} \frac{1}{v_k(T)} \left(y_k + \sum_{j \in T} z_j \right).$$

In other words, for each k , one needs to find the set T which minimizes $(y_k + \sum_{j \in T} z_j)/v_k(T)$. If y_k is interpreted as a fixed cost incurred by agent k and z_j as the cost of item j , then T is the set that minimizes the ratio of cost relative to value. For a simple-minded bidder who is interested in the items in a subset T_0 and no other item, i.e., $v_k(T) = v_k(T_0)$ if $T_0 \subseteq T$ and $v_k(T) = 0$, otherwise, T_0 is the minimizer. Another simple case is additive valuations, i.e., $v_k(T) = \sum_{j \in T} a_j^k$, where $a_j^k \geq 0$ is the value of item j for agent k . In this situation, $\frac{1}{v_k(T)} (y_k + \sum_{j \in T} z_j) \leq \beta$ for a set T and a positive real β if and only if $\sum_{j \in T} (\beta a_j^k - z_j) \geq y_k$ and hence the minimal β for which such a set T exists is readily determined by binary search on β .

2 A Fast Algorithm for Convex Decompositions

Let $x^* \in \mathcal{Q}$ be arbitrary. Carr and Vempala [CV02] showed how to construct a convex combination of points in $\mathcal{Q}_{\mathcal{I}}$ dominating αx^* using a polynomial number of calls to an α -integrality-gap-verifier

for $\mathcal{Q}_{\mathcal{I}}$. Lavi and Swamy [LS11] modified the construction to get an exact convex decomposition $\alpha x^* = \sum_{i \in \mathcal{N}} \lambda_i x^i$ for the case of packing linear programs. The construction uses the Ellipsoid method. We show an approximate version that replaces the use of the Ellipsoid method by the multiplicative weights update (MWU) method. For any $\varepsilon > 0$, we show how to obtain a convex decomposition of $\alpha x^*/(1 + \varepsilon)$. Let s be the number of non-zero components of x^* . The size of the decomposition and the number of calls to the α -integrality gap verifier are $O(s\varepsilon^{-2} \ln s)$.

This section is structured as follows. We first review Khandekar's FPTAS for covering linear programs (Subsection 2.1). We then use it and the α -integrality gap verifier to construct, on input $x^* \in \mathcal{Q}$, a dominating convex combination for $\alpha x^*/(1 + 4\varepsilon)$ (Subsection 2.2). In Subsection 2.3, we show how to convert a dominating convex combination into an exact convex decomposition. Finally, in Subsection 2.4, we put the pieces together.

2.1 Khandekar's Algorithm for Covering Linear Programs

Consider a covering linear program:

$$\min c^T x \quad \text{subject to} \quad Ax \geq b, \quad x \geq 0, \quad (2)$$

where $A \in \mathbb{R}_{\geq 0}^{m \times n}$ is an $m \times n$ matrix with non-negative entries and $c \in \mathbb{R}_{\geq 0}^n$ and $b \in \mathbb{R}_{\geq 0}^m$ are non-negative vectors. We assume the availability of a κ -approximation oracle for some $\kappa \in (0, 1]$.

$\mathcal{O}_{\kappa}(z)$: Given $z \in \mathbb{R}_{\geq 0}^m$, the oracle finds a column j of A that maximizes $\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i}$ within a factor of κ :

$$\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i} \geq \kappa \cdot \max_{j' \in [n]} \frac{1}{c_{j'}} \sum_{i=1}^m \frac{z_i a_{ij'}}{b_i}$$

For an exact oracle $\kappa = 1$, Khandekar [Kha04] gave an algorithm which computes a feasible solution \hat{x} to (2) such that $c^T \hat{x} \leq (1 + 4\varepsilon)z^*$ where z^* is the value of an optimal solution. The algorithm makes $O(m\varepsilon^{-2} \log m)$ calls to the oracle, where m is the number of rows in A . There are algorithms predating Khandekar's work, see, for example, [K97, Chapter 4]

Theorem 3 (Generalization of Khandekar's algorithm to arbitrary $\kappa \leq 1$). *Let $\varepsilon \in (0, \frac{1}{2}]$ and let z^* be the value of an optimum solution to (2). Procedure $\text{Covering}(\mathcal{O}_{\kappa})$ (see Algorithm 3 in Appendix I) terminates in at most $m\lceil \varepsilon^{-2} \ln m \rceil$ iterations with a feasible solution \hat{x} of (2) of at most $m\lceil \varepsilon^{-2} \ln m \rceil$ positive components. At termination, it holds that*

$$c^T \hat{x} \leq \frac{(1 + 4\varepsilon)}{\kappa} z^*. \quad (3)$$

For completeness, we give a proof of Khandekar's result in Appendix I. The proof of Theorem 3 can be modified to give (see Appendix I):

Corollary 1. *Suppose $b = \mathbf{1}$, $c = \mathbf{1}$, and we use the following oracle \mathcal{O}' instead of \mathcal{O} in Algorithm 3:*

$\mathcal{O}'(z)$: *Given $z \in \mathbb{R}_{\geq 0}^m$, such that $\mathbf{1}^T z = 1$, the oracle finds a column j of A such that $z^T A \mathbf{1}_j \geq 1$.*

Then the algorithm terminates in at most $m\lceil \varepsilon^{-2} \ln m \rceil$ iterations with a feasible solution \hat{x} having at most $m\lceil \varepsilon^{-2} \ln m \rceil$ positive components, such that $\mathbf{1}^T \hat{x} \leq 1 + 4\varepsilon$.

2.2 Finding a Dominating Convex Combination

Recall that we use \mathcal{N} to index the elements in $\mathcal{Q}_{\mathcal{I}}$. We assume the availability of an α -integrality-gap-verifier \mathcal{F} for $\mathcal{Q}_{\mathcal{I}}$. We will use the results of the preceding section and show how to obtain for any $x^* \in \mathcal{Q}$ and any positive ε a convex composition of points in $\mathcal{Q}_{\mathcal{I}}$ that covers $\alpha x^*/(1 + 4\varepsilon)$. Our algorithm requires $O(s\varepsilon^{-2} \ln s)$ calls to the oracle, where s is size of the support of x^* .

Theorem 4. Let $\varepsilon > 0$ be arbitrary. Given a fractional point $x^* \in \mathcal{Q}$ and an α -integrality-gap verifier \mathcal{F} for $\mathcal{Q}_{\mathcal{I}}$, we can find a convex combination \bar{x} of integral points in $\mathcal{Q}_{\mathcal{I}}$ such that

$$\frac{\alpha}{1+4\varepsilon} \cdot x^* \leq \bar{x} = \sum_{i \in \mathcal{N}} \lambda_i x^i.$$

The convex decomposition has size at most $s \lceil \varepsilon^{-2} \ln s \rceil$, where s is the number of positive entries of x^* . The algorithm makes at most $s \lceil \varepsilon^{-2} \ln s \rceil$ calls to the integrality-gap verifier.

Proof. The task of finding the multipliers λ_i is naturally formulated as a covering LP ([CV02]), namely,

$$\begin{aligned} \min \quad & \sum_{i \in \mathcal{N}} \lambda_i \\ \text{s.t.} \quad & \sum_{i \in \mathcal{N}} \lambda_i x_j^i \geq \alpha \cdot x_j^* \quad \text{for all } j, \\ & \sum_{i \in \mathcal{N}} \lambda_i \geq 1, \quad \lambda_i \geq 0. \\ & \lambda_i \geq 0. \end{aligned} \tag{4}$$

Clearly, we can restrict our attention to the $j \in S^+ := \{j : x_j^* > 0\}$ and rewrite the constraint for $j \in S^+$ as $\sum_{i \in \mathcal{N}} \lambda_i x_j^i / (\alpha x_j^*) \geq 1$. For simplicity of notation, we assume $S^+ = [1..s]$. We thus have a covering linear program as in (2) with $m := s + 1$ constraints, $n := |\mathcal{N}|$ variables λ_i , right-hand side $b := \mathbf{1}$, cost vector $c := \mathbf{1}$, and constraint matrix $A = (a_{j,i})$ (note that we use j for the row index and i for the column index), where

$$a_{j,i} := \begin{cases} x_j^i / (\alpha x_j^*) & 1 \leq j \leq s, i \in \mathcal{N} \\ 1 & j = s + 1, i \in \mathcal{N} \end{cases}$$

Thus we can apply Corollary 1 of Section 2.1, provided we can efficiently implement the required oracle \mathcal{O}' . We do so using \mathcal{F} .

Oracle \mathcal{O}' has is given a \tilde{z} such that $1^T \tilde{z} = 1$. Let us conveniently write $\tilde{z} = (w, z)$, where $w \in \mathbb{R}_{\geq 0}^s$, $z \in \mathbb{R}_{\geq 0}$, and $\sum_{j=1}^s w_j + z = 1$. Oracle \mathcal{O}' needs to find a column i such that $\tilde{z}^T A \mathbf{1}_i \geq 1$. In our case $\tilde{z}^T A \mathbf{1}_i = \sum_{j=1}^s w_j x_j^i / \alpha x_j^* + z$, and we need to find a column i for which this expression is at least one. Since z does not depend on i , we concentrate on the first term. Define

$$V_j := \begin{cases} \frac{w_j}{\alpha x_j^*} & \text{for } j \in S^+ \\ 0 & \text{otherwise.} \end{cases}$$

Call algorithm \mathcal{F} with $x^* \in \mathcal{Q}$ and $V := (V_1, \dots)$. \mathcal{F} returns an integer solution $x^i \in \mathcal{Q}_{\mathcal{I}}$ such that

$$\sum_{j \in S^+} \frac{w_j}{\alpha x_j^*} x_j^i = V^T x^i \geq \alpha \cdot V^T x^* = \sum_{j \in S^+} w_j,$$

and hence,

$$\sum_{j \in S^+} \frac{w_j}{\alpha x_j^*} x_j^i + z \geq \sum_{j \in S^+} w_j + z = 1.$$

Thus i is the desired column of A .

It follows by Corollary 1 that Algorithm 3 finds a feasible solution $\lambda' \in \mathbb{R}_{\geq 0}^{|\mathcal{N}|}$ to the covering LP (4), and a set $\mathcal{Q}'_{\mathcal{I}} \subseteq \mathcal{Q}_{\mathcal{I}}$ of vectors (returned by \mathcal{F}), such that $\lambda'_i > 0$ only for $i \in \mathcal{N}'$, where \mathcal{N}' is the index set returned by oracle \mathcal{O}' and $|\mathcal{N}'| \leq s \lceil \varepsilon^{-2} \ln s \rceil$. Also $\Lambda := \sum_{i \in \mathcal{N}'} \lambda'_i \leq (1 + 4\varepsilon)$. Scaling λ'_i by Λ , we obtain a set of multipliers $\{\lambda_i = \lambda'_i / \Lambda : i \in \mathcal{N}'\}$, such that $\sum_{i \in \mathcal{N}'} \lambda_i = 1$ and

$$\sum_{i \in \mathcal{N}'} \lambda_i x^i \geq \frac{\alpha}{1+4\varepsilon} x^*.$$

We may assume $x_j^i = 0$ for all $j \notin S^+$ whenever $\lambda_i > 0$; otherwise simply replace x^i by a vector in which all components not in S^+ are set to zero. By the packing property this is possible. \square

Algorithm 1 Changing a dominating convex decomposition into an exact decomposition

Require: A packing convex set \mathcal{Q} and point $x^* \in \mathcal{Q}$ and a convex combination $\sum_{i \in \mathcal{N}} \lambda_i x^i$ of integral points in $\mathcal{Q}_{\mathcal{I}}$ dominating x^* .

Ensure: A convex decomposition $x^* = \sum_{i \in \mathcal{N}'} \lambda_i x^i$ with $x^i \in \mathcal{Q}_{\mathcal{I}}$.

```
1: while  $\Delta_j := \sum_{i \in \mathcal{N}} \lambda_i x^i - x_j^* > 0$  for some  $j$  do
2:   let  $i$  be such that  $\lambda_i x_j^i > 0$  and  $\Delta_j > 0$  for some  $j$ .
3:   if there is a  $j$  such that  $\lambda_i x_j^i > 0$  and  $\sum_{h \in \mathcal{N}} \lambda_h x_h^i - \lambda_i \mathbf{1}_j \geq x^*$  then
4:     replace  $x^i$  by  $x^i - \mathbf{1}_j$ .
5:   else
6:     Among the indices  $j$  with  $x_j^i > 0$  and  $\Delta_j > 0$ , let  $k$  minimize  $\Delta_k / x_k^i$ .
7:     let  $y$  be such that  $y_j = x_j^i$ , if  $\Delta_j = 0$ , and  $y_j = 0$ , if  $\Delta_j > 0$ .
8:     change the lefthand side of (5) as follows: replace  $\lambda_i$  by  $\lambda_i - \Delta_k / x_k^i$  and increase the coefficient of  $y$  by  $\Delta_k / x_k^i$ .
9:   end if
10: end while
```

2.3 From Dominating Convex Combination to Exact Convex Decomposition

We will show how to turn a dominating convex combination into an exact decomposition. The construction is general and uses only the packing property. Such a construction seems to have been observed in [LS05], but was not made explicit. Kraft, Fadaei, and Bichler [KFB14] describe an alternative construction. Their construction may increase the size of the convex decomposition (= number of non-zero λ_i) by a multiplicative factor s and an additive factor s^2 . In contrast, our construction increases the size only by an additive factor s .

Theorem 5. *Let $x^* \in \mathcal{Q}$ be dominated by a convex combination $\sum_{i \in \mathcal{N}} \lambda_i x^i$ of integral points in $\mathcal{Q}_{\mathcal{I}}$, i.e.,*

$$\sum_{i \in \mathcal{N}} \lambda_i x^i \geq x^*. \quad (5)$$

Then Algorithm 1 achieves equality in (5). It increases the size of the convex combination by at most s , where s is the number of positive components of x^ .*

Proof. Let $S^+ = \{j : x_j^* > 0\}$. We may assume $x_j^i = 0$ for all $j \notin S^+$ and all $i \in \mathcal{N}$ with $\lambda_i > 0$.

For $j \in S^+$, let $\Delta_j = \sum_{i \in \mathcal{N}} \lambda_i x_j^i - x_j^*$ be the gap in the j -th component. If $\Delta_j = 0$ for all $j \in S^+$, we are done. Otherwise, choose j and $i \in \mathcal{N}$ such that $\Delta_j > 0$ and $\lambda_i x_j^i > 0$.

Let $\mathbf{1}_j$ be the j -th unit vector. If, for some j with $x_j^i > 0$ and $\Delta_j > 0$, replacing x^i by $x^i - \mathbf{1}_j$ maintains feasibility, i.e., satisfies constraint (5), we perform this replacement. Since x^i is an integer vector in $\mathcal{Q}_{\mathcal{I}}$, the vector $x^i - \mathbf{1}_j$ is nonnegative and, by the packing property, in $\mathcal{Q}_{\mathcal{I}}$. The replacement decreases Δ_j by λ_i and does not increase the number of nonzero λ_i .

Otherwise, $\Delta_j < \lambda_i$ for all j with $\Delta_j > 0$ and $x_j^i > 0$. Since x^i is integral, we also have $\Delta_j \leq \lambda_i x_j^i$ for all such j . Among the indices j with $\Delta_j > 0$ and $x_j^i > 0$, let k minimize Δ_k / x_k^i . Let y be such that $y_j = x_j^i$ if $\Delta_j = 0$ and $y_j = 0$ if $\Delta_j > 0$. Then $y \in \mathcal{Q}_{\mathcal{I}}$ since \mathcal{Q} is a packing polytope. In the convex combination, replace

$$\lambda_i x^i \quad \text{by} \quad \left(\lambda_i - \frac{\Delta_k}{x_k^i}\right) \cdot x^i + \frac{\Delta_k}{x_k^i} \cdot y.$$

Notice that $\lambda_i - \frac{\Delta_k}{x_k^i} \geq 0$. Let Δ'_j be the new gaps. Then clearly $\Delta'_j = \Delta_j$, if $\Delta_j = 0$. Consider any j with $\Delta_j > 0$. Then

$$\Delta'_j = \Delta_j - \frac{\Delta_k}{x_k^i} \cdot x_j^i = \begin{cases} 0 & \text{if } j = k \\ \geq (\Delta_j - \frac{\Delta_k}{x_k^i}) \cdot x_j^i = 0 & \text{if } j \neq k. \end{cases}$$

The inequality in the second case holds since $\Delta_k/x_k^i \leq \Delta_j/x_j^i$. We have decreased the number of nonzero Δ_j by one at the cost one additional nonzero λ_i . Thus the total number of vectors added to the convex decomposition is at most s . \square

2.4 Fast Convex Decomposition

We are now ready to prove Theorem 1.

Proof of Theorem 1. Theorem 4 yields a convex combination of integer points of \mathcal{Q}_I dominating $\alpha x^*/(1 + 4\epsilon)$. The convex decomposition has size at most $s \lceil \epsilon^{-2} \ln s \rceil$, where s is the number of positive entries of x^* . The algorithm makes at most $s \lceil \epsilon^{-2} \ln s \rceil$ calls to the integrality-gap verifier. Theorem 5 turns this dominating convex combination into an exact combination. It adds up to s additional vectors to the convex combination. \square

3 Approximately Truthful-in-Expectation Mechanisms

The goal of this section is to derive an approximate VCG-mechanism. We do not longer assume that the fractional SWM-problem can be solved exactly, but instead assume that we have an FPTAS for it. We will first design a randomized fractional algorithm (Theorem 6 in Subsection 3.1) and then convert the fractional mechanism into an integral mechanism and prove Theorem 2 in Subsection 3.2.

3.1 Approximately Truthful-in-Expectation Fractional Mechanisms

Theorem 6. *Let $\epsilon_0 \in (0, 1/2]$. Define $\epsilon = \Theta(\frac{\epsilon_0^5}{n^4})$. Assuming that the fractional SWM-problem has an FPTAS, is separable, and has a dominant allocation for every player i , there is a polynomial time randomized fractional mechanism (Algorithm 2) with the following properties:*

- (D1) *No positive transfer, i.e., prices are nonnegative.*
- (D2) *Individually rational with probability $1 - \epsilon_0$, i.e., the utility of any truth-telling player is non-negative with probability at least $1 - \epsilon_0$.*
- (D3) *$(1 - \epsilon_0)$ -truthful-in-expectation, i.e., reporting the truth maximizes the expected utility of a player up to a factor $1 - \epsilon_0$.*
- (D4) *γ -socially efficient, where $\gamma = (1 - \epsilon)(1 - \epsilon_0)$.*

In order to present Algorithm 2 and prove Theorem 6, we introduce some notation and prove some preliminary Lemmas. Let

$$L_i := \sum_{j \neq i} v_j(u^j) \quad \text{and} \quad \beta_i := \epsilon L_i. \quad (6)$$

Note that L_i does not depend on the valuation of player i . Let \mathcal{A} be an ϵ -approximation algorithm for the LP relaxation of SWM. Note that \mathcal{A} is polynomial time since the running time of an FPTAS is polynomial in $\frac{1}{\epsilon}$. We use $\mathcal{A}(v)$ to denote the outcome of \mathcal{A} on input v ; $\mathcal{A}(v)$ is a fractional allocation in \mathcal{Q} . In the following, we will apply \mathcal{A} to different valuations which we denote by $v = (v_i, v_{-i})$, $\bar{v} = (\bar{v}_i, v_{-i})$, and $v' = (\mathbf{0}, v_{-i})$. Here v_i is the reported valuation of player i , \bar{v}_i is his true valuation and $v'_i = \mathbf{0}$. We denote the allocation returned by \mathcal{A} on input v (resp., \bar{v} , v') by x (resp., \bar{x} , x'). Note that x , \bar{x} , x' are fractional allocations.

We first bound the maximal change in social welfare induced by a change of the valuation of the i -th player.

Lemma 1. *Let $\epsilon \geq 0$ and let \mathcal{A} be an ϵ -approximation algorithm which returns allocation x on input vector v . Let $\hat{x} \in \mathcal{Q}$ be an arbitrary point, then*

$$v(x) \geq v(\hat{x}) - \beta_i - \epsilon \cdot v_i(\hat{x}) \quad (7)$$

for every i .

Algorithm 2 The mechanism M of Theorem 6. The vectors u^i are defined as in (1) and the quantities L_i are defined in (6). The definitions of q_0, q_j , active and inactive player are given in the proof of Theorem 6.

Require: A valuation vector v , a packing convex set \mathcal{Q} and an ε -approximation algorithm, where ε is as Theorem 6.

Ensure: An allocation $x \in \mathcal{Q}$ and a payment $p \in \mathbb{R}^n$ satisfying (D1) to (D4).

- 1: Choose an index $j \in \{0, 1, \dots, n\}$, where 0 is chosen with probability q_0 and $j \in \{1, \dots, n\}$ is chosen with probability $q_j = (1 - q_0)/n$.
- 2: **if** $j = 0$ **then**
- 3: Use ε -approximation algorithm \mathcal{A} to compute an allocation $x = (x_1, \dots, x_n) \in \mathcal{Q}$ and compute payments with payment rule (8). For all inactive i , change x_i and p_i to zero.
- 4: **else**
- 5: For every $1 \leq i \leq n$, set

$$\begin{cases} x_i = u^i, p_i = \eta' L_i & \text{if } i = j \text{ and } i \text{ is active,} \\ x_i = u^i, p_i = 0 & \text{if } i = j \text{ and } i \text{ is inactive,} \\ x_i = 0, p_i = 0 & \text{if } i \neq j. \end{cases}$$

- 6: **end if**
 - 7: **return** (x, p)
-

Proof. We have

$$\begin{aligned} v(x) &\geq (1 - \varepsilon) \max_{x \in \mathcal{Q}} v(x) \\ &\geq (1 - \varepsilon) v(\hat{x}) \\ &= v(\hat{x}) - \varepsilon \cdot \sum_{j \neq i} v_j(\hat{x}) - \varepsilon \cdot v_i(\hat{x}) \\ &\geq v(\hat{x}) - \beta_i - \varepsilon \cdot v_i(\hat{x}), \end{aligned}$$

where the first inequality follows from the fact that \mathcal{A} is an ε -approximation algorithm, and the last inequality follows from $\varepsilon \sum_{j \neq i} v_j(\hat{x}) \leq \varepsilon \sum_{j \neq i} v_j(u^j) = \beta_i$. \square

We use the following payment rule:

$$p_i(v) := \max\{p_i^{VCG}(v) - \beta_i, 0\} \quad (8)$$

where

$$p_i^{VCG}(v) := v_{-i}(x') - v_{-i}(x).$$

$v_{-i}(x) = \sum_{j \neq i} v_j(x)$, $x = \mathcal{A}(v)$ and $x' = \mathcal{A}(0, v_{-i})$. Observe the similarity in the definition of $p_i^{VCG}(v)$ to the VCG payment rule. In both cases, the payment is defined as the difference of the total value of two allocations to the players different from i . The first allocation ignores the influence of player i ($x' = \mathcal{A}(0, v_{-i})$) and the second allocation takes it into account ($x = \mathcal{A}(v)$). The difference to the VCG rule is that x' and x are not true maximizers but are computed by an ε -approximation algorithm.

Let $U_i(v) = \bar{v}_i(x) - p_i(v)$ be the utility of player i for bid vector v . Note that the value of the allocation $x = \mathcal{A}(v)$ is evaluated with the true valuation \bar{v}_i of player i . Let $U_i(\bar{v}) = \bar{v}_i(\bar{x}) - p_i(\bar{v})$ be the utility of player i for valuation vector $\bar{v} = (\bar{v}_i, v_{-i})$.

Lemma 2. Let $\varepsilon \geq 0$ and let \mathcal{A} be an ε -approximation algorithm. Let M_0 be the mechanism with allocation function $\mathcal{A}(v)$ and the payment rule (8). M_0 is an individually rational mechanism with no positive transfer, such that for all i ,

$$U_i(\bar{v}) \geq U_i(v) - \varepsilon \cdot \bar{v}_i(x) - 3\beta_i. \quad (9)$$

Proof. By definition, $p_i(v) \geq 0$ for all v and all x ; so the mechanism has no positive transfer. We next address individual rationality. Assume $p_i(\bar{v}) = p_i^{VCG}(\bar{v}) - \beta_i > 0$, as otherwise $U_i(\bar{v}) \geq 0$. We have

$$\begin{aligned} U_i(\bar{v}) &= \bar{v}_i(\bar{x}) - p_i(\bar{v}) \\ &= \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) + \beta_i \\ &= \bar{v}_i(\bar{x}) + \bar{v}_{-i}(\bar{x}) - \bar{v}_{-i}(x') + \beta_i \\ &= \bar{v}(\bar{x}) - \bar{v}(x') + \bar{v}_i(x') + \beta_i \\ &\geq (1 - \varepsilon)\bar{v}_i(x') \geq 0, \end{aligned}$$

where the first inequality follows from Lemma 1 with $v = \bar{v}$ and $\hat{x} = x'$.

Finally, we prove (9). We have $v'(x') = v_{-i}(x')$, $v'(x) = v_{-i}(x)$, and $v'_i(x) = 0$. Thus,

$$p_i^{VCG}(v) = v_{-i}(x') - v_{-i}(x) = v'(x') - v'(x) + \varepsilon \cdot v'_i(x)$$

Applying Lemma 1 for $v = v'$ and $\hat{x} = x$, we obtain

$$v'(x') - v'(x) + \varepsilon \cdot v'_i(x) \geq -\beta_i$$

Therefore,

$$p_i^{VCG}(v) + \beta_i \geq 0. \quad (10)$$

To see (9), we consider two cases:

Case 1: $p_i(v) = 0$. Then using (10)

$$U_i(\bar{v}) = \bar{v}_i(\bar{x}) - 0 \geq \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i.$$

Case 2: $p_i(v) = p_i^{VCG}(v) - \beta_i$.

$$U_i(\bar{v}) = \bar{v}_i(\bar{x}) - p_i(\bar{v}) = \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) + \beta_i \geq \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i,$$

where the last inequality follows from $\beta_i \geq 0$. Therefore, in both cases we have:

$$U_i(\bar{v}) \geq \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i.$$

Now by using the definition of p_i^{VCG} and Lemma 1, we get

$$\begin{aligned} U_i(\bar{v}) &\geq \bar{v}_i(\bar{x}) - p_i^{VCG}(\bar{v}) - \beta_i \\ &= \bar{v}_i(\bar{x}) + \bar{v}_{-i}(\bar{x}) - \bar{v}_{-i}(x') - \beta_i \\ &= \bar{v}(\bar{x}) - \bar{v}_{-i}(x') - \beta_i \\ &\geq \bar{v}(x) - \beta_i - \varepsilon \bar{v}_i(x) - \bar{v}_{-i}(x') - \beta_i \\ &= \bar{v}_i(x) - p_i^{VCG}(v) - \varepsilon \bar{v}_i(x) - 2\beta_i \\ &\geq \bar{v}_i(x) - p_i(v) - \beta_i - \varepsilon \bar{v}_i(x) - 2\beta_i \\ &= U_i(v) - \varepsilon \bar{v}_i(x) - 3\beta_i. \end{aligned}$$

□

In what follows we prove Theorem 6.

Proof of Theorem 6. Define $q_0 = (1 - \frac{\varepsilon_0}{n})^n$, $\bar{\varepsilon} = \varepsilon_0/2$, and $q_j = (1 - q_0)/n$ for $1 \leq j \leq n$. Let $\eta = \bar{\varepsilon}(1 - q_0)^2/n^3$, $\eta' = \eta/q_j$, and $\varepsilon = \eta\bar{\varepsilon}(1 - q_0)/(8n)$. Then using⁷ $q_0 = (1 - \frac{\varepsilon_0}{n})^n \geq 1 - \varepsilon_0$ and $q_0 = (1 - \frac{\varepsilon_0}{n})^n \leq 1 - \varepsilon_0/2$, we get

$$\frac{\varepsilon_0^5}{128n^4} = \frac{\bar{\varepsilon}^2(\varepsilon_0/2)^3}{8n^4} \leq \varepsilon = \eta\bar{\varepsilon}(1 - q_0)/(8n) = \frac{\bar{\varepsilon}^2(1 - q_0)^3}{8n^4} \leq \frac{\bar{\varepsilon}^2\varepsilon_0^3}{8n^4} = \frac{\varepsilon_0^5}{16n^4},$$

⁷ Let $f(x) = (1 - x/n)^n - 1 + x$. Then $f'(x) = (1 - x/n)^{n-1} + 1 \geq 0$. Hence, for $0 \leq x \leq 1$ and $n \geq 1$, the function is increasing and $f(x) \geq f(0) = 0$.

as stated in the Theorem. Let $U_i(v) = \bar{v}_i(x) - p_i(v)$ be the utility of player i obtained by the mechanism M_0 of Lemma 2. Let further $\hat{U}_i(v) = v_i(x) - p_i(v)$. Following [DRVY11], we call player i *active* if the following two conditions hold:

$$\hat{U}_i(v) + \frac{\bar{\varepsilon}q_i}{q_0}v_i(u^i) \geq \frac{q_i}{q_0}\eta' L_i, \quad (11)$$

$$v_i(u^i) \geq \eta L_i. \quad (12)$$

Note that these conditions do not depend on the true valuation \bar{v} . We denote by $T = T(v)$ the set of active players when the valuation is $v = (v_1, \dots, v_n)$. Note that L_i does not depend on v_i . Thus when we refer to conditions (11) and (12) for \bar{v} , we replace v and x by \bar{v} and \bar{x} on the left side and keep the right side unchanged. Non-negativity of payments is immediate from the definition of mechanism M and Lemma 2. Moreover, the utility of a truth-telling bidder i can be negative only if he/she is allocated in step 5, i.e., at most with probability q_i . It follows that the mechanism is individually rational with probability at least $1 - \sum_{i=1}^n q_i = q_0 = (1 - \frac{\varepsilon_0}{n})^n \geq 1 - \varepsilon_0$.

Now we address truthfulness. Let us denote the expected utility of player i obtained from the mechanism in Algorithm 2 on input $v \in \mathcal{V}$ by $\mathbb{E}[U'_i(v)]$. Assume $j = 0$ in Algorithm 2. We run ε -approximation algorithm \mathcal{A} on v to compute allocation $x = (x_1, \dots, x_n)$. Then we change x_i and p_i to zero for all inactive i . Let \tilde{x} be the allocation obtained in this way. The value for player i is $v_i(\tilde{x})$. When the i -th player is active, this value is equal to $v_i(x)$ because v_i depends only on the valuation in the i -th group (separability property). Therefore in this case his utility is $U_i(v)$. So we have that

$$\mathbb{E}[U'_i(v)] = \begin{cases} q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \eta' L_i) & \text{if } i \in T(v), \\ q_i \bar{v}_i(u^i) & \text{if } i \notin T(v). \end{cases} \quad (13)$$

We first observe

$$\mathbb{E}[U'_i(\bar{v})] \geq (1 - \bar{\varepsilon})q_i \cdot \bar{v}_i(u^i). \quad (14)$$

Indeed, the inequality is trivially satisfied if $i \notin T(\bar{v})$. On the other hand, if $i \in T(\bar{v})$, then (11) implies $U_i(\bar{v}) = \hat{U}_i(\bar{v}) \geq \frac{q_i}{q_0}(\eta' L_i - \bar{\varepsilon} \bar{v}_i(u^i))$, therefore

$$\begin{aligned} \mathbb{E}[U'_i(\bar{v})] &= q_0 \cdot U_i(\bar{v}) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &\geq q_0 \cdot \frac{q_i}{q_0}(\eta' L_i - \bar{\varepsilon} \bar{v}_i(u^i)) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &= (1 - \bar{\varepsilon})q_i \cdot \bar{v}_i(u^i). \end{aligned}$$

We now consider four cases:

Case 1: $i \in T(\bar{v}) \cap T(v)$. Note that (12) for \bar{v} implies $\beta_i = \varepsilon L_i \leq \frac{\varepsilon \bar{v}_i(u^i)}{\eta}$. Thus, by Lemma 2, and using assumption (1) that $\bar{v}_i(x) \leq \bar{v}_i(u^i)$, we have

$$U_i(\bar{v}) \geq U_i(v) - \varepsilon(1 + \frac{3}{\eta})\bar{v}_i(u^i) \geq U_i(v) - \frac{4\varepsilon}{\eta}\bar{v}_i(u^i). \quad (15)$$

Hence by using (13) and (15), we have

$$\begin{aligned} \mathbb{E}[U'_i(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &\leq q_0(U_i(\bar{v}) + \frac{4\varepsilon}{\eta}\bar{v}_i(u^i)) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &= \underbrace{q_0 U_i(\bar{v}) + q_i(\bar{v}_i(u^i) - \eta' L_i)}_{\mathbb{E}[U'_i(\bar{v})]} + q_0 \frac{4\varepsilon}{\eta} \bar{v}_i(u^i) \\ &= \mathbb{E}[U'_i(\bar{v})] + q_0 \frac{4\varepsilon}{\eta} \bar{v}_i(u^i). \end{aligned}$$

Now applying (14) in the above inequality, we get

$$\begin{aligned}\mathbb{E}[U'_i(v)] &\leq \mathbb{E}[U'_i(\bar{v})] + q_0 \frac{4\varepsilon}{\eta} \bar{v}_i(u^i) \\ &\leq \left(1 + \frac{q_0}{(1-\bar{\varepsilon})q_i} \frac{4\varepsilon}{\eta}\right) \mathbb{E}[U'_i(\bar{v})] \\ &\leq (1 + \bar{\varepsilon}) \mathbb{E}[U'_i(\bar{v})],\end{aligned}$$

where the last inequality follows from the definition of ε . Note that (since $q_0 \leq 1$ and $\bar{\varepsilon} \leq 1/2$)

$$\varepsilon \frac{q_0}{(1-\bar{\varepsilon})q_i} \frac{4}{\eta} \leq \varepsilon \frac{1}{(1-\bar{\varepsilon})q_i} \frac{4}{\eta} \leq \frac{\eta \bar{\varepsilon} (1-q_0)}{8n} \frac{8}{q_i \eta} = \bar{\varepsilon}.$$

Case 2: $i \notin T(v)$. By (14), we have

$$\mathbb{E}[U'_i(v)] = q_i \bar{v}_i(u^i) \leq \frac{1}{1-\bar{\varepsilon}} \mathbb{E}[U'_i(\bar{v})] \leq (1 + \varepsilon_0) \mathbb{E}[U'_i(\bar{v})].$$

Since, $\frac{1}{1-\bar{\varepsilon}} = 1 + \bar{\varepsilon}(1 + \bar{\varepsilon} + \bar{\varepsilon}^2 + \dots) \leq 1 + 2\bar{\varepsilon} = 1 + \varepsilon_0$.

Case 3: $i \in T(v) \setminus T(\bar{v})$ and (12) does not hold for \bar{v} . Since $U_i(v) \leq \bar{v}_i(u^i)$, we have

$$\begin{aligned}\mathbb{E}[U'_i(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \eta' L_i) \leq (q_0 + q_i) \bar{v}_i(u^i) - q_i \eta' L_i < (q_0 + q_i - 1) \bar{v}_i(u^i) \\ &\leq 0 \leq q_i \bar{v}_i(u^i) = \mathbb{E}[U'_i(\bar{v})],\end{aligned}$$

where the second inequality holds because (12) does not hold for \bar{v} and $q_i \eta' / \eta = 1$.

Case 4: $i \in T(v) \setminus T(\bar{v})$ and (12) holds for \bar{v} . Then (11) does not hold for \bar{v} and hence

$$U_i(\bar{v}) = \hat{U}_i(\bar{v}) < \frac{q_i}{q_0} (\eta' L_i - \bar{\varepsilon} \bar{v}_i(u^i)). \quad (16)$$

Since (12) holds for \bar{v} , we have (15). Hence by (14), (15) and (16) we have

$$\begin{aligned}\mathbb{E}[U'_i(v)] &= q_0 \cdot U_i(v) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &\leq q_0 \left(U_i(\bar{v}) + \frac{4\varepsilon}{\eta} \bar{v}_i(u^i) \right) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &\leq q_i \eta' L_i - q_i \bar{\varepsilon} \bar{v}_i(u^i) + \frac{4\varepsilon}{\eta} \bar{v}_i(u^i) + q_i(\bar{v}_i(u^i) - \eta' L_i) \\ &= (1 - \bar{\varepsilon}) q_i \cdot \bar{v}_i(u^i) + \frac{4\varepsilon}{\eta} \bar{v}_i(u^i) \\ &\leq \left(1 + \frac{1}{(1-\bar{\varepsilon})q_i} \frac{4\varepsilon}{\eta}\right) \mathbb{E}[U'_i(\bar{v})] \\ &\leq (1 + \bar{\varepsilon}) \mathbb{E}[U'_i(\bar{v})],\end{aligned}$$

where the last inequality follows from the definition of ε (see Case 1).

We finally argue about the approximation ratio. Note that for $i \notin T(v)$, one of the inequalities (11) or (12) does not hold. Also, $U_i(v) \geq 0$ in this case since $p_i = 0$, and hence $v_i(u^i) < \max\{\eta, \eta'/\bar{\varepsilon}\} L_i = \eta' L_i / \bar{\varepsilon}$. Since \mathcal{A} returns allocation x that is $(1 - \varepsilon)$ -social efficiency

and⁸ $q_i - q_0 n \frac{\eta'}{\varepsilon} \geq 0$, it follows that for any $v \in \mathcal{V}$, (recall $x = \mathcal{A}(v)$)

$$\begin{aligned}
\mathbb{E}[v(\tilde{x})] &= q_0 \sum_{i \in T(v)} v_i(x) + \sum_{i \in [n]} q_i v_i(u^i) \\
&= q_0 \sum_{i \in [n]} v_i(x) - q_0 \sum_{i \notin T(v)} v_i(x) + \sum_{i \in [n]} q_i v_i(u^i) \\
&= q_0 v(x) - q_0 \sum_{i \notin T(v)} v_i(u^i) + \sum_{i \in [n]} q_i v_i(u^i) \\
&> q_0 v(x) - q_0 \frac{\eta'}{\varepsilon} \sum_{i \notin T(v)} L_i + \sum_{i \in [n]} q_i v_i(u^i) \\
&= q_0 v(x) - q_0 \frac{\eta'}{\varepsilon} \sum_{i \notin T(v)} \sum_{j \neq i} v_j(u^j) + \sum_{i \in [n]} q_i v_i(u^i) \\
&\geq q_0 v(x) - q_0 \frac{\eta'}{\varepsilon} n \sum_{j \in [n]} v_j(u^j) + \sum_{i \in [n]} q_i v_i(u^i) \\
&\geq q_0 v(x) + \sum_{i \in [n]} \left(q_i - q_0 n \frac{\eta'}{\varepsilon} \right) v_i(u^i) \\
&\geq q_0 (1 - \varepsilon) \cdot \max_{z \in Q} v(z) \\
&\geq (1 - \varepsilon_0)(1 - \varepsilon) \cdot \max_{z \in Q} v(z).
\end{aligned}$$

□

3.2 Approximately Truthful-in-Expectation Integral Mechanisms

In this subsection, we derive a randomized mechanism M' which returns an integral allocation. Let $\varepsilon > 0$ be arbitrary. First run Algorithm 2 to obtain x and $p(v)$. Then compute a convex decomposition of $\frac{\alpha}{1+4\varepsilon}x$, which is $\frac{\alpha}{1+4\varepsilon}x = \sum_{j \in \mathcal{N}} \lambda_j^x x^j$. Finally with probability λ_j^x (we use the superscript x to distinguish the convex decompositions of different x) return the allocation x^j and charge the i -th player the price $p_i(v) \frac{v_i(x^j)}{v_i(x)}$, if $v_i(x) > 0$, and zero otherwise. We now prove Theorem 2.

Proof of Theorem 2. Let M be a fractional randomized mechanism obtained in Theorem 6. Since M has no positive transfer, M' does neither. M is individually rational with probability $1 - \varepsilon_0$, therefore for any allocation \bar{x} , we have $\bar{v}_i(\bar{x}) - p_i(\bar{v}) \geq 0$ with probability $1 - \varepsilon_0$. So

$$\bar{v}_i(x^l) - p_i(\bar{v}) \frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})} = (\bar{v}_i(\bar{x}) - p_i(\bar{v})) \frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})} \geq 0,$$

hence M' is individually rational with probability $1 - \varepsilon_0$. Now we prove truthfulness. Let $\mathbb{E}[U_i''(\bar{v})]$ be the expected utility of player i when she inputs her true valuation and let $\mathbb{E}[U_i''(v)]$ be her

⁸ $q_0 n \frac{\eta'}{\varepsilon} \leq \frac{n\eta}{q_i \varepsilon} = \frac{n^2 \varepsilon (1-q_0)^2}{(1-q_0) \varepsilon n^3} = \frac{1-q_0}{n} = q_i$.

expected utility when she inputs v_i . Then by definition of $\mathbb{E}[U_i''(\bar{v})]$, we have

$$\begin{aligned}
\mathbb{E}[U_i''(\bar{v})] &= \mathbb{E}_{\bar{x} \sim M(\bar{v})} \left[\sum_{l \in \mathcal{N}} \lambda_l^{\bar{x}} \left(\bar{v}_i(x^l) - p_i(\bar{v}) \frac{\bar{v}_i(x^l)}{\bar{v}_i(\bar{x})} \right) \right] \\
&= \mathbb{E}_{\bar{x} \sim M(\bar{v})} \left[\left(\bar{v}_i \left(\sum_{l \in \mathcal{N}} \lambda_l^{\bar{x}} x^l \right) - p_i(\bar{v}) \frac{\bar{v}_i(\sum_{l \in \mathcal{N}} \lambda_l^{\bar{x}} x^l)}{\bar{v}_i(\bar{x})} \right) \right] \\
&= \mathbb{E}_{\bar{x} \sim M(\bar{v})} \left[\frac{\alpha}{1+4\varepsilon} \bar{v}_i(\bar{x}) - \frac{\alpha}{1+4\varepsilon} p_i(\bar{v}) \right] \\
&= \frac{\alpha}{1+4\varepsilon} \mathbb{E}_{\bar{x} \sim M(\bar{v})} [\bar{v}_i(\bar{x}) - p_i(\bar{v})] \\
&= \frac{\alpha}{1+4\varepsilon} \mathbb{E}[U'(\bar{v})] \\
&\geq (1-\varepsilon_0) \frac{\alpha}{1+4\varepsilon} \mathbb{E}[U'(v)] \\
&= (1-\varepsilon_0) \frac{\alpha}{1+4\varepsilon} \mathbb{E}_{x \sim M(v)} [\bar{v}(x) - p_i(v)] \\
&= (1-\varepsilon_0) \mathbb{E}_{x \sim M(v)} \left[\frac{\alpha}{1+4\varepsilon} \bar{v}(x) - p_i(v) \frac{\alpha}{1+4\varepsilon} \cdot \frac{v_i(x)}{v_i(x)} \right] \\
&= (1-\varepsilon_0) \mathbb{E} \left[\sum_{l \in \mathcal{N}} \lambda_l^x \left(\bar{v}_i(x^l) - p_i(v) \frac{v_i(x^l)}{v_i(x)} \right) \right] \\
&= (1-\varepsilon_0) \mathbb{E}[U_i'(v)].
\end{aligned}$$

Taking expectation with respect to x shows that the mechanism is $\frac{\alpha(1-\varepsilon_0)(1-\varepsilon)}{1+4\varepsilon}$ -socially efficient.

$$\begin{aligned}
\mathbb{E}[v(x)] &= \mathbb{E}_{x \sim M(v)} \left[\sum_{l \in \mathcal{N}} \lambda_l^x v(x^l) \right] = \mathbb{E}_{x \sim M(v)} \left[v \left(\sum_{l \in \mathcal{N}} \lambda_l^x x^l \right) \right] = \mathbb{E}_{x \sim M(v)} \left[v \left(\frac{\alpha}{1+4\varepsilon} x \right) \right] \\
&= \frac{\alpha}{1+4\varepsilon} \mathbb{E}_{x \sim M(v)} [v(x)] \geq \frac{\alpha}{1+4\varepsilon} (1-\varepsilon_0)(1-\varepsilon) \max_{z \in Q} v(z).
\end{aligned}$$

This completes the proof of Theorem 2. □

References

- [AHK12] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.
- [BI06] D. Bienstock and G. Iyengar. Approximating fractional packings and coverings in $O(1/\epsilon)$ iterations. *SIAM J. Comput.*, 35(4):825–854, 2006.
- [BKV05] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. In *STOC*, pages 39–48, 2005.
- [CEF10] G. Christodoulou, K. Elbassioni, and M. Fouz. Truthful mechanisms for exhibitions. In *WINE*, pages 170–181, 2010.
- [CV02] Robert D. Carr and Santosh Vempala. Randomized metarounding. *Random Struct. Algorithms*, 20(3):343–352, 2002.
- [DRVY11] Shaddin Dughmi, Tim Roughgarden, Jan Vondrák, and Qiqi Yan. An approximately truthful-in-expectation mechanism for combinatorial auctions using value queries. *CoRR*, abs/1109.1053, 2011.
- [EJ15] K. Elbassioni and M. Jha. On the power of combinatorial bidding in web display ads. In *ICIW*, pages 46–55, 2015.
- [EMR15] Khaled Elbassioni, Kurt Mehlhorn, and Fahimeh Ramezani. Towards More Practical Linear Programming-Based Techniques for Algorithmic Mechanism Design. In Martin Hoefer, editor, *Algorithmic Game Theory*, volume 9347 of *Lecture Notes in Computer Science*, pages 98–109. Springer Berlin Heidelberg, 2015.

- [GK95] M.D. Grigoriadis and L.G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53 – 58, 1995.
- [GK07] N. Garg and J. Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM J. Comput.*, 37(2):630–652, 2007. a preliminary version appeared in FOCS 1998.
- [GLS88] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, New York, 1988.
- [HKV11] M. Hoefer, T. Kesselheim, and B. Vöcking. Approximation algorithms for secondary spectrum auctions. In *SPAA*, pages 177–186, 2011.
- [K97] J. Könemann. Fast combinatorial algorithms for packing and covering problems. Master’s thesis, Fachbereich Informatik, Universität des Saarlandes, 1997.
- [KFB14] D. Kraft, S. Fadaei, and M. Bichler. Fast convex decomposition for truthful social welfare approximation. In *WINE*, pages 120–132, 2014.
- [Kha04] R. Khandekar. *Lagrangian Relaxation Based Algorithms for convex Programming Problems*. PhD thesis, Indian Institute of Technology, Delhi, India, 2004.
- [KS98] S. Kolliopoulos and C. Stein. Approximating disjoint-path problems using greedy algorithms and packing integer programs. In *IPCO*, pages 153–168, 1998.
- [KY07] C. Koufogiannakis and N.E. Young. Beating simplex for fractional packing and covering linear programs. In *FOCS*, pages 494–504, 2007.
- [LS05] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. In *FOCS*, pages 595–604, 2005.
- [LS11] Ron Lavi and Chaitanya Swamy. Truthful and near-optimal mechanism design via linear programming. *J. ACM*, 58(6):25, 2011.
- [NRTV07] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [PST91] S.A. Plotkin, D.B. Shmoys, and É. Tardos. Fast approximation algorithms for fractional packing and covering problems. In *FOCS*, pages 495–504, 1991.
- [Rag88] P. Raghavan. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *J. Comput. Syst. Sci.*, 37(2):130–143, 1988.
- [WRM15] Di Wang, Satish Rao, and Michael W. Mahoney. Unified acceleration method for packing and covering problems via diameter reduction. *CoRR*, abs/1508.02439, 2015.
- [You01] N.E. Young. Sequential and parallel algorithms for mixed packing and covering. In *FOCS*, pages 538–546, 2001.
- [ZO15] Z. Allen Zhu and L. Orecchia. Nearly-linear time positive LP solver with faster convergence rate. In *STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 229–236, 2015.

Appendix I: Khandekar’s Algorithm for Covering Linear Programs

Consider a covering linear program:

$$\min c^T x \quad \text{subject to} \quad Ax \geq b, \quad x \geq 0, \quad (17)$$

where $A \in \mathbb{R}_{\geq 0}^{m \times n}$ is an $m \times n$ matrix with non-negative entries and $c \in \mathbb{R}_{\geq 0}^n$ and $b \in \mathbb{R}_{\geq 0}^m$ are non-negative vectors. We assume the availability of a κ -approximation oracle for some $\kappa \in (0, 1]$.

$\mathcal{O}_\kappa(z)$: Given $z \in \mathbb{R}_{\geq 0}^m$, the oracle finds a column j of A that maximizes $\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i}$ within a factor of κ :

$$\frac{1}{c_j} \sum_{i=1}^m \frac{z_i a_{ij}}{b_i} \geq \kappa \cdot \max_{j' \in [n]} \frac{1}{c_{j'}} \sum_{i=1}^m \frac{z_i a_{ij'}}{b_i} \quad (18)$$

We use A_i to denote the i -th row of A . Algorithm 3 constructs vectors $x(t) \in \mathbb{R}_{\geq 0}^n$, for $t = 0, 1, \dots$, until $M(t) := \min_{i \in [m]} A_i x(t)/b_i$ becomes at least $T := \frac{\ln m}{\varepsilon^2}$. Define the *active list* at time t by $L(t) := \{i \in [m] : A_i x(t-1)/b_i < T\}$. For $i \in L(t)$, define

$$p_i(t) := (1 - \varepsilon)^{A_i x(t-1)/b_i}, \quad (19)$$

Algorithm 3 Covering(\mathcal{O}_κ)

Require: a covering system (A, b, c) given by a κ -approximation oracle \mathcal{O}_κ , where $A \in \mathbb{R}_{\geq 0}^{m \times n}$, $b \in \mathbb{R}_{> 0}^m$, $c \in \mathbb{R}_{> 0}^n$, and an accuracy parameter $\varepsilon \in (0, 1/2]$

Ensure: A feasible solution $\hat{x} \in \mathbb{R}_{\geq 0}^n$ to (2) s.t. $c^T \hat{x} \leq \frac{(1+4\varepsilon)}{\kappa} z^*$

```

1:  $x(0) := 0$ ;  $t := 0$ ; and  $T := \frac{\ln m}{\varepsilon^2}$ 
2: while  $M(t) < T$  do
3:    $t := t + 1$ 
4:   Let  $j(t) := \mathcal{O}_\kappa(p(t)/\|p(t)\|_1)$ 
5:    $x_{j(t)}(t) := x_{j(t)}(t-1) + \delta(t)$  and  $x_j(t) = x_j(t-1)$  for  $j \neq j(t)$ 
6: end while
7: return  $\hat{x} = \frac{x(t)}{M(t)}$ 

```

and set $p_i(t) = 0$ for $i \notin L(t)$. At each time t , the algorithm calls the oracle with the vector $z_t = p(t)/\|p(t)\|_1$, and increases the variable $x_{j(t)}$ by

$$\delta(t) := \min_{i \in L(t) \text{ and } a_{i,j(t)} \neq 0} \frac{b_i}{a_{i,j(t)}}, \quad (20)$$

where $j(t)$ is the index returned by the oracle.

Proof of Theorem 3. Note that the RHS of (18) is positive for our choice of z_t since every row of A contains a non-zero entry and hence $\sum_{i \in L(t)} p_i(t) a_{ij(t)} / (b_i c_{j(t)}) > 0$. This conclude that there exist at least one $i \in L(t)$ which $a_{i,j(t)}$ is non zero and thus $\delta(t) > 0$ always. In each iteration, some entry of x is increased and hence the values $A_i x(t)/b_i$ are non-decreasing. Thus $L(t+1) \subseteq L(t)$ for all t . At the end, we scale $x(t)$ by $M(t)$ to guarantee feasibility.

Let $\mathbf{1}_j$ denote the j -th unit vector of dimension n and $B \in \mathbb{R}^{m \times m}$ be a diagonal matrix with entries $b_{ii} = b_i$. Feasibility is obvious since we scale by $M(t)$. The bound on the number of iterations is also obvious since in each iteration at least one of the $A_i x/b_i$ increases by one and we remove i from the active list once $A_i x/b_i$ reaches T . We conclude that the number of iterations is bounded by $m \lceil T \rceil$. Let t_0 be the number of iterations, i.e., vectors $x(0), x(1), \dots, x(t_0)$ are defined and $M(t_0 - 1) < T \leq M(t_0)$. In the t -th iteration exactly one entry of x is increased by $\delta(t)$ and hence $\mathbf{1}^T x(t_0) = \sum_{1 \leq t \leq t_0} \delta(t)$ and $A_i x(t)/b_i \leq A_i x(t-1)/b_i + 1$ for $i \in L(t)$. To show (3), we analyze the decrease of $\|p(t)\|_1$. Let $t \leq t_0$. Then

$$\begin{aligned}
\sum_{i \in L(t)} (1 - \varepsilon)^{A_i x(t)/b_i} &= \sum_{i \in L(t)} (1 - \varepsilon)^{A_i x(t-1)/b_i + \delta(t) A_i \mathbf{1}_{j(t)}/b_i} \\
&= \sum_{i \in L(t)} p_i(t) (1 - \varepsilon)^{\delta(t) A_i \mathbf{1}_{j(t)}/b_i} \\
&\leq \sum_{i \in L(t)} p_i(t) (1 - \varepsilon \delta(t) A_i \mathbf{1}_{j(t)}/b_i) \\
&\quad (\text{using (20) conclude that } \delta(t) A_i \mathbf{1}_{j(t)}/b_i \leq 1 \text{ and} \\
&\quad (1 - \varepsilon)^x \leq 1 - \varepsilon x \text{ for all } \varepsilon \in [0, 1), x \in [0, 1]) \\
&= \|p(t)\|_1 \left(1 - \frac{\varepsilon \delta(t) p(t)^T B^{-1} A \mathbf{1}_{j(t)}}{\|p(t)\|_1} \right) \\
&\leq \|p(t)\|_1 e^{-\varepsilon \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}} \quad \text{since } 1 - x \leq e^{-x}. \quad (21)
\end{aligned}$$

By using $L(t+1) \subseteq L(t)$, we have

$$\begin{aligned}\|p(t+1)\|_1 &= \sum_{i \in L(t+1)} (1-\varepsilon)^{A_i x(t)/b_i} \\ &\leq \sum_{i \in L(t)} (1-\varepsilon)^{A_i x(t)/b_i}\end{aligned}\tag{22}$$

and hence applying inequalities (21) and (22) we get,

$$\|p(t+1)\|_1 \leq \sum_{i \in L(t)} (1-\varepsilon)^{A_i x(t)/b_i} \leq \|p(t)\|_1 e^{-\varepsilon \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}}.\tag{23}$$

Let $i_0 \in L(t_0)$ be arbitrary. Then

$$\begin{aligned}(1-\varepsilon)^{A_{i_0} x(t_0)/b_{i_0}} &\leq \sum_{i \in L(t_0)} (1-\varepsilon)^{A_i x(t_0)/b_i} \\ &\leq \|p(t_0)\|_1 e^{-\varepsilon \delta(t_0) \frac{p(t_0)^T}{\|p(t_0)\|_1} B^{-1} A \mathbf{1}_{j(t_0)}} \\ &\leq \|p(0)\|_1 e^{-\varepsilon \sum_{1 \leq t \leq t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}},\end{aligned}$$

where the second inequality uses (21) for $t = t_0$ and the third inequality uses (23) for $0 \leq t < t_0$. Taking logs and using $\|p(0)\|_1 = m$, we conclude that

$$A_{i_0} x(t_0)/b_{i_0} \cdot \ln(1-\varepsilon) \leq \ln m - \varepsilon \sum_{1 \leq t \leq t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}\tag{24}$$

We next relate the objective value $c^T x(t_0) = \sum_{1 \leq t \leq t_0} c_j(t) \delta(t)$ at time t_0 to the optimal value z^* by the following claim.

Claim. $\sum_{1 \leq t \leq t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)} \geq \frac{\kappa \cdot c^T x(t_0)}{z^*}.$

Proof. Let $x^* \in \mathbb{R}_{\geq 0}^n$ be an optimal solution to (2). Since x^* is feasible, $B^{-1} A x^* \geq \mathbf{1}$, and thus for any t ,

$$p(t)^T B^{-1} A x^* \geq p(t)^T \mathbf{1} = \|p(t)\|_1.$$

By the choice of the index $j(t)$, we have that $\frac{1}{c_j(t)} p(t)^T B^{-1} A \mathbf{1}_{j(t)} \geq \frac{1}{c_j} \kappa p(t)^T B^{-1} A \mathbf{1}_j$ for all $j \in [n]$. Since $z^* = c^T x^*$, we conclude further

$$\begin{aligned}z^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} &= \sum_{j \in [n]} c_j x_j^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} \\ &= \sum_{j \in [n]} c_j \cdot \frac{c_j(t)}{c_j(t)} x_j^* p(t)^T B^{-1} A \mathbf{1}_{j(t)} \\ &\geq \sum_{j \in [n]} c_j \cdot \frac{c_j(t)}{c_j} x_j^* \kappa p(t)^T B^{-1} A \mathbf{1}_j \\ &= \kappa c_{j(t)} p(t)^T B^{-1} A x^* \\ &\geq \kappa c_{j(t)} \|p(t)\|_1.\end{aligned}$$

Multiplying both sides of this inequality by $\delta(t)/\|p(t)\|_1$ and summing up over $1 \leq t \leq t_0$ finishes the proof of the claim. \square

Using the claim above, we deduce from (24)

$$A_{i_0} x(t_0)/b_{i_0} \cdot \ln(1-\varepsilon) \leq \ln m - \varepsilon \cdot \frac{\kappa \cdot c^T x(t_0)}{z^*}$$

Dividing both sides by $M(t_0)$, arranging, and using $M(t_0) \geq T = (\ln m)/\varepsilon^2$, $A_{i_0}x(t_0)/b_{i_0} \leq A_{i_0}x(t_0 - 1)/b_{i_0} + 1 \leq T + 1$, and $\frac{\ln \frac{1}{1-\varepsilon}}{\varepsilon} \leq 1 + 2\varepsilon$, valid for all $\varepsilon \in (0, \frac{1}{2}]$, we obtain

$$\begin{aligned} \frac{\kappa \cdot c^T \hat{x}}{z^*} &= \frac{\kappa \cdot c^T x(t_0)}{M(t_0)z^*} \leq \frac{\ln \frac{1}{1-\varepsilon}}{\varepsilon} \cdot \frac{A_{i_0}x(t_0)/b_{i_0}}{M(t_0)} + \frac{\ln m}{\varepsilon \cdot M(t_0)} \\ &\leq (1 + 2\varepsilon) \frac{T + 1}{T} + \varepsilon \leq 1 + 4\varepsilon. \end{aligned}$$

□

Proof of Corollary 1. Recall (24):

$$A_{i_0}x(t_0)/b_{i_0} \cdot \ln(1 - \varepsilon) \leq \ln m - \varepsilon \sum_{1 \leq t \leq t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} B^{-1} A \mathbf{1}_{j(t)}.$$

With assumption $b = \mathbf{1}$, we have,

$$A_{i_0}x(t_0) \cdot \ln(1 - \varepsilon) \leq \ln m - \varepsilon \sum_{1 \leq t \leq t_0} \delta(t) \frac{p(t)^T}{\|p(t)\|_1} A \mathbf{1}_{j(t)}.$$

The vector $z_t = p(t)/\|p(t)\|_1$ satisfies $\mathbf{1}^T z_t = 1$. Apply oracle \mathcal{O}' with input vector z_t , it returns index $j(t)$ such that we have $\frac{p(t)^T}{\|p(t)\|_1} A \mathbf{1}_{j(t)} \geq 1$. Thus, we have

$$A_{i_0}x(t_0) \cdot \ln(1 - \varepsilon) \leq \ln m - \varepsilon \cdot \mathbf{1}^T x(t_0).$$

Proceeding as in the proof of Theorem 3, we get the result.

□